NoSQL

DataBase Management System

목차

- NoSQL이란?
- 기존 RDBMS와의 차이
- NoSQL의 종류와 장,단점
- 추천 NoSQL DB

NoSQL이란?

- 기존 관계형 데이터베이스(RDBMS)와는 다른 형태의 DB
- Not only SQL, non SQL, Non-relational
- 다양한 데이터베이스 모델 존재
- 대부분의 데이터 구조가 명확하지 않음

RDBMS vs NoSQL

Relational Database	NoSQL
Scale-up - 서버 한대 중심으로 확장	Scale-out - 여러대의 서버를 중심으로 확장
무결성	유연성
데이터 중복 제거	데이터 중복 허용
트랜잭션	빠른 쓰기, 읽기

- RDBMS - 데이터들의 구조가 변경이 없고,
  테이블 간의 관계가 명확할 때.
- NoSQL - 정확한 데이터 구조를 알 수 없고,
  많은 양의 데이터를 처리해야할 때.

https://db-engines.com/en/ranking

선정기준:

검색엔진
검색 횟수,
SNS,
구인 사이트

Rank			DBMS	Database Model	Score		
Sep 2022	Aug 2022	Sep 2021			Sep 2022	Aug 2022	Sep 2021
1.	1.	1.	Oracle	Relational, Multi-model	1238.25	-22.54	-33.29
2.	2.	2.	MySQL	Relational, Multi-model	1212.47	+9.61	-0.06
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	926.30	-18.66	-44.55
4.	4.	4.	PostgreSQL	Relational, Multi-model	620.46	+2.46	+42.95
5.	5.	5.	MongoDB	Document, Multi-model	489.64	+11.97	-6.87
6.	6.	6.	Redis	Key-value, Multi-model	181.47	+5.08	+9.53
7.	↑ 8.	↑ 8.	Elasticsearch	Search engine, Multi-model	151.44	-3.64	-8.80
8.	↓ 7.	↓ 7.	IBM Db2	Relational, Multi-model	151.39	-5.83	-15.16
9.	9.	↑ 11.	Microsoft Access	Relational	140.03	-6.47	+23.09
10.	10.	↓ 9.	SQLite	Relational	138.82	-0.05	+10.17
11.	11.	↓ 10.	Cassandra	Wide column	119.11	+0.97	+0.12
12.	12.	12.	MariaDB	Relational, Multi-model	110.16	-3.74	+9.46
13.	13.	↑ 21.	Snowflake	Relational	103.50	+0.38	+51.43
14.	14.	↓ 13.	Splunk	Search engine	94.05	-3.39	+2.45
15.	15.	↑ 16.	Amazon DynamoDB	Multi-model	87.42	+0.16	+10.49
16.	16.	↓ 15.	Microsoft Azure SQL Database	Relational, Multi-model	84.42	-1.75	+6.16
17.	17.	↓ 14.	Hive	Relational	78.43	-0.22	-7.14
18.	18.	↓ 17.	Teradata	Relational, Multi-model	66.58	-2.49	-3.09
19.	19.	↓ 18.	Neo4j	Graph	59.48	+0.12	+1.85
20.	↑ 22.		Databricks	Multi-model	55.62	+1.00	
21.	↓ 20.	↑ 22.	Solr	Search engine, Multi-model	54.05	-1.73	+4.24
22.	↓ 21.	↓ 19.	SAP HANA	Relational, Multi-model	51.82	-3.14	-4.42
23.	23.	↓ 20.	FileMaker	Relational	51.57	-1.55	-0.75
24.	24.	↑ 25.	Google BigQuery	Relational	50.12	+0.09	+6.20
25.	25.	↓ 23.	SAP Adaptive Server	Relational, Multi-model	42.85	-1.96	-4.16

MongoDB - Document DB

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

ID: breakfast

```
{
        "type": "toast",
        "bread": "whole wheat",
        "spread": [
                "butter",
                "jam"
        ]
}
```

ID: lunch

```
{
        "type": "salad",
        "vegetarian": false,
        "ingredients": [
                "spinach",
                "tomato",
                "cucumber",
                "carrot",
                "dressing": [
                        "olive oil",
                        "vinegar",
                        "honey",
                        "lemon",
                        "salt",
                        "pepper",
                ],
                "tuna",
                "walnuts"
        ],
        "rating": "5 stars",
        "restaurant": "Skylight Diner"
}
```

ID: dinner

```
{
        "type": "pizza",
        "size": "large",
        "toppings": [
                "pepperoni",
                "tomato",
                "sausage"
        ],
        "price": 9.00,
        "presliced": true
}
```

1. 테이블 구조, 칼럼 명, 칼럼 개수 와 같은
   데이터 구조가 명확하지않음.
2. 값에 해당하는 부분이 xml이나 json과 같은 문서로 저장

MongoDB - Document DB

Relational

ID	first_name	last_name	cell	city	year_of_birth	location_x	location_y
1	'Mary'	'Jones'	'516-555-2048'	'Long Island'	1986	'-73.9876'	'40.7574'

ID	user_id	profession
10	1	'Developer'
11	1	'Engineer'

ID	user_id	name	version
20	1	'MyApp'	1.0.4
21	1	'DocFinder'	2.5.7

ID	user_id	make	year
30	1	'Bentley'	1973
31	1	'Rolls Royce'	1965

MongoDB

```
{
first_name: "Mary",
last_name: "Jones",
cell: "516-555-2048",
city: "Long Island",
year_of_birth: 1986,
location: {
        type: "Point",
        coordinates: [-73.9876, 40.7574]
},
profession: ["Developer", "Engineer"],
apps: [
 { name: "MyApp",
   version: 1.0.4 },
 { name: "DocFinder",
   version: 2.5.7 }
 ],
cars: [
   { make: "Bentley",
     year: 1973 },
   { make: "Rolls Royce",
     year: 1965 }
 ]
}
```

다른 테이블을 활용하는 JOIN을 하지 않는다.

MongoDB – 장단점

장점

1. JOIN을 사용하지 않아 빠른 연산 가능
2. 비정형 데이터에 적합

단점

1. 데이터 중복 발생 가능
2. 저장공간 낭비 발생
3. 무결성을 입증하기 어려움

Redis – key-value DB

- 오직 key-value만을 위한 간단한 DB
- 값에 대한 모든 데이터 타입 허용 (문자, 숫자, 리스트, 배열)
- key로만 검색 가능

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Redis – 장단점

장점

1. 단순한 구조로 빠른 조회
2. 쉬운 구현
3. 다양한 형태의 데이터 입력

단점

1. 값을 기준으로 검색 불가
2. SQL과 같은 질의언어 존재x

ElasticSearch - Search engine DB

- 형태소 분석
- 검색 순위 제공
- 검색에 알맞은 최상의 형태로
  데이터를 갖춤(역색인)

ElasticSearch – 장단점

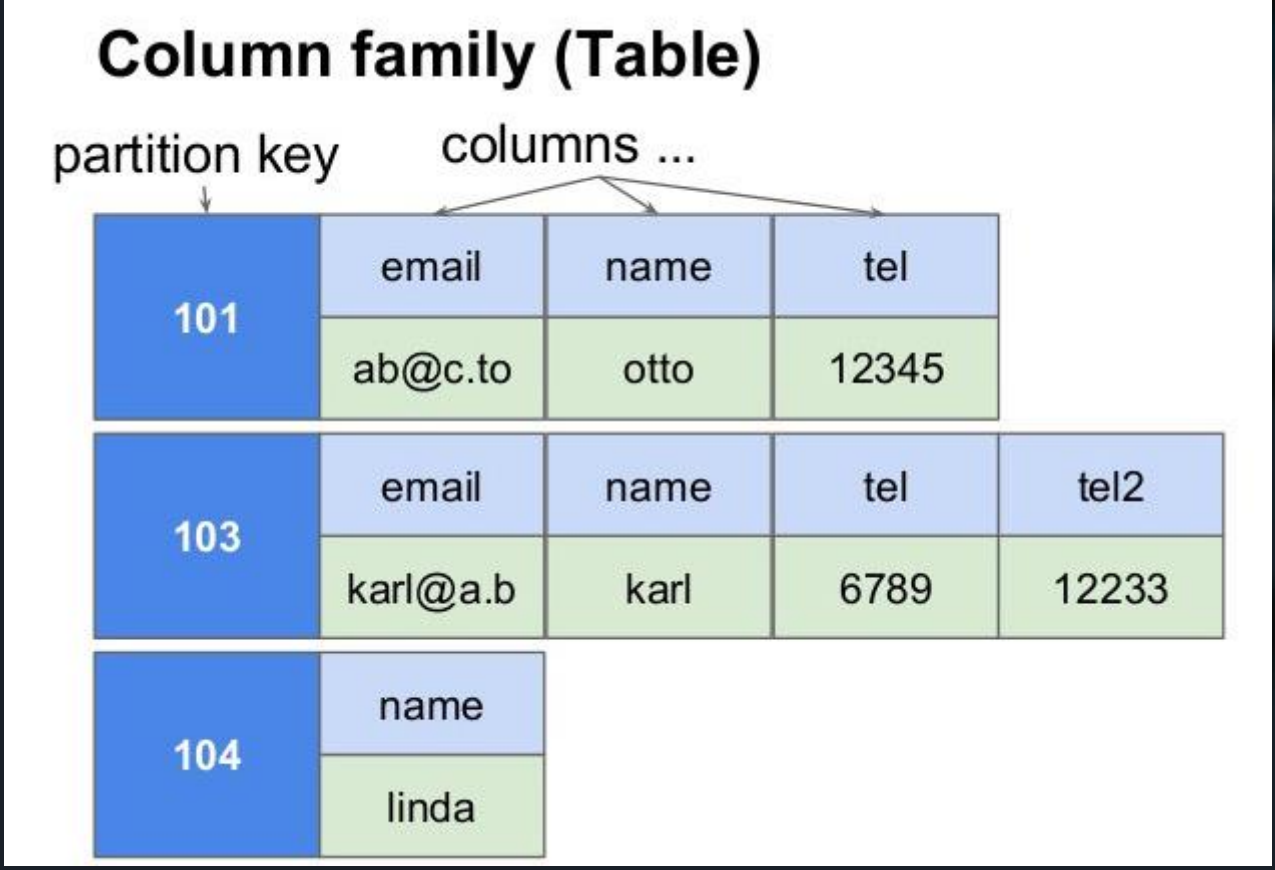
장점

1. 역색인을 통해 빠른 속도로 조회 가능

단점

1. 복잡한 조건 검색 불가
2. 트랜잭션 불가
3. 쓰기 속도가 느림

Cassandra - Column-oriented DB



- 필수 칼럼은 파티션으로서 정의한다.
- 칼럼에 대한 제한이 없다.

Cassandra - Column-oriented DB

사번	1001
이름	홍길동
급여	1000
전화	09)123-2134
사번	1002
이름	임꺽정
급여	2000
전화	09)253-9378
	...

사번	1001
사번	1002
사번	1003
사번	1004
이름	홍길동
이름	임꺽정
이름	이동철
이름	이철수
	...

- 로우가 아닌 칼럼 단위로 데이터를 저장한다.

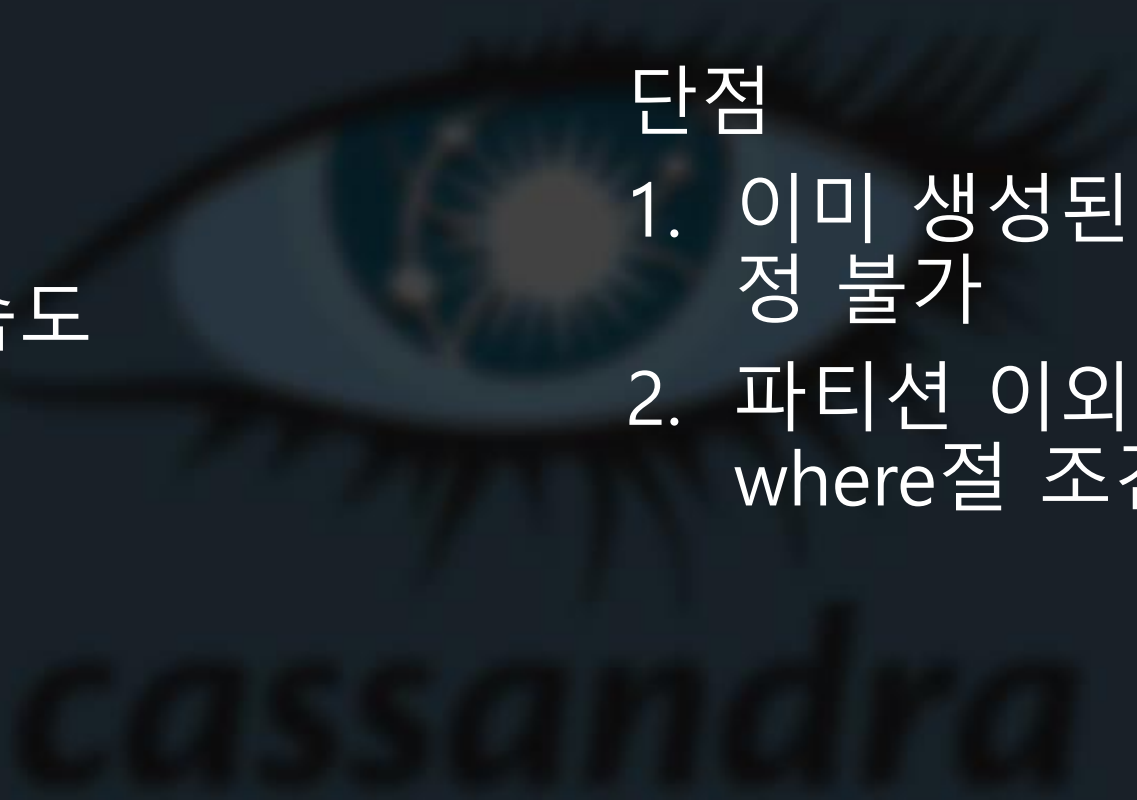
Cassandra – 장단점

장점

1. 높은 압축률
2. 빠른 쓰기 속도

단점

1. 이미 생성된 파티션을 수정 불가
2. 파티션 이외의 칼럼으로 where절 조건처리 어려움



추천 로깅용 DB - Cassandra

```
[14:18:03.570][http-nio-8080-exec-7][I][ADMIN](ServiceConnector):status = 1
[14:18:07.510][http-nio-8080-exec-9][Q][ADMIN](SepoaXmlParser):----------
[14:18:07.510][http-nio-8080-exec-9][Q][ADMIN](SepoaXmlParser):- Method Name : doQuery
[14:18:07.510][http-nio-8080-exec-9][Q][ADMIN](SepoaXmlParser):- XML Path : D:/Dev/03.WorkSpace/Poa-RPA/p
[14:18:07.510][http-nio-8080-exec-9][Q][ADMIN](SepoaXmlParser):----------
[14:18:07.512][http-nio-8080-exec-9][M][ADMIN](SC_010):recvData[0]=000
[14:18:07.512][http-nio-8080-exec-9][M][ADMIN](SC_010):recvData[1]=1000
[14:18:07.512][http-nio-8080-exec-9][M][ADMIN](SC_010):recvData[2]=
[14:18:07.512][http-nio-8080-exec-9][Q][ADMIN](SC_010):setString[1]='000'
[14:18:07.512][http-nio-8080-exec-9][Q][ADMIN](SC_010):setString[2]='1000'
[14:18:07.574][http-nio-8
[MariaDB:jdbc:mariadb://1
SELECT
   STORE_CODE
  , ( SELECT SBUPR.SELLE
   WHERE SBUPR.HOUSE_COD
    AND SBUPR.COMPANY_CO
    AND SBUPR.BIZ_CODE =
  , OWNER_CODE
  , BIZ_CODE
  , ITEM_EA
  , ITEM_NAME
  , ORDER_DATE
  , TER_CODE
  , ORDER_CODE
  , ITEM_BOX
  , BIZ_NAME
  , ITEM_TOTAL_AMOUNT
  , ITEM_ORIGIN_AMOUNT
  , ITEM_VAT_AMOUNT
  , ORDER_TIME
```

```
16:12:43.467][http-nio-8080-exec-4][Q][ADMIN](SC_001):formatting(milliseconds) = 0
16:12:43.526][http-nio-8080-exec-4][Q][ADMIN](SepoaSQLManager):SQLManager option change : [Lo
16:12:43.570][http-nio-8080-exec-4][I][ADMIN](ServiceConnector):sepoa.svc.sc.SC_001.doQuery
16:12:43.570][http-nio-8080-exec-4][I][ADMIN](ServiceConnector):response time = 227ms
16:12:43.570][http-nio-8080-exec-4][I][ADMIN](ServiceConnector):message = null
16:12:43.570][http-nio-8080-exec-4][I][ADMIN](ServiceConnector):status = 1
16:12:45.722][pool-1-thread-1][E][ADMIN](SC_001):
java.lang.reflect.InvocationTargetException: null
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at sepoa.svc.scraping.ScrapingWrapper.start(ScrapingWrapper.java:16)
    at sepoa.svc.sc.SC_001.run(SC_001.java:177)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:748)
Caused by: java.io.FileNotFoundException: D:\Dev\03.WorkSpace\Poa-RPA\poarpa\gs\hodme.js (지정
    at java.io.FileInputStream.open0(Native Method)
    at java.io.FileInputStream.open(FileInputStream.java:195)
    at java.io.FileInputStream.<init>(FileInputStream.java:138)
    at java.io.FileInputStream.<init>(FileInputStream.java:93)
    at java.io.FileReader.<init>(FileReader.java:58)
    at sepoa.svc.scraping.gs.GS.setLoginPost(GS.java:153)
    at sepoa.svc.scraping.Scraping.getLogin(Scraping.java:114)
    at sepoa.svc.scraping.Scraping.start(Scraping.java:58)
    ... 9 common frames omitted
```

```
INSERT into log
  (status,responseTime,executeTime,message,xmlPath,SQL)
  VALUES ('0','235ms','162438','성공적으로 ..','SELECT ..')
```

```
INSERT into log
  (status,responseTime,executeTime,message,exception,)
  VALUES ('0','235ms','162438','java.lang..','FileNo..')
```

- 데이터베이스를 이용한 빠른 조회
- 빠른 입력
- 정해진 데이터 유형이 없어 서로 다른 메뉴에서 간편하게 사용 가능
- 익숙한 쿼리(CQL)